Developing a domain-specific plug-in for a modelling platform: the good, the bad, the ugly

Developing a Domain-Specific Plug-In for a Modelling Platform: The Good, the Bad, the Ugly

Lionel Montrieux Yijun Yu Michel Wermelinger
Centre for Research in Computing
The Open University, Milton Keynes, UK
{Lionel.Montrieux, Yijun.Yu, Michel.Wermelinger}@open.ac.uk

***Abstract*—Domain-Specific Modelling Languages (DSML) allow software engineers to use the techniques and tools of Model-Driven Engineering (MDE) to express, represent and analyse a particular domain. By defining DSMLs as UML profiles, i.e. domain-specific extensions of the UML metamodel, development time for DSMLs can be greatly reduced by extending existing UML tools. In this paper, we reflect on our own experience in building rbacUML, a DSML for Role-Based Access Control modelling and analysis, as a plugin for a UML modelling platform. We describe what motivated our choice, and discuss the advantages and drawbacks of using an existing platform to develop a DSML on top of UML and additional analysis tooling.**

***Index Terms*—MDE, RBAC, OCL, Eclipse, Modelling, Plugin**

I. Introduction

The last decade or two have seen tremendous amounts of research conducted on Model-Driven Engineering (MDE), a software engineering paradigm that advocates the incremental transformation of models of a software to be built into more detailed models, until code is eventually produced. Arguably the most well-known MDE framework is Model-Driven Architecture (MDA), which includes UML as a modelling language.

If modelling languages such as UML can be seen as the modelling equivalent of general-purpose programming languages, then Domain-Specific Modelling Languages (DSML) are the modelling equivalent of Domain-Specific Languages (DSL). There has however been less research into DSMLs and how they can be used as first class citizens in a MDE approach. Our work tries to fill that gap in the security context, in particular for Role-Based Access Control (RBAC).

To illustrate our research and conduct experiments, we are developing `rbacUML`, a plugin that not only defines a DSML for RBAC-based authorisation using UML's extension mechanism, but also allows designers to model access control requirements and to verify models against those requirements.

In this paper we reflect on our experience in building `rbacUML` as a plugin. We discuss how non-functional requirements such as access control can be tightly integrated into an MDE approach, and point out the benefits and challenges of using the platform's capabilities to provide advanced MDE features.

The rest of this paper is organised as follows: Section II gives essential background on MDE and RBAC. Then, Section III describes the `rbacUML` approach and the philosophy that guided the choices made during its development. Section IV discusses our choice to develop a plugin, while Section V focuses on the implementation of the `rbacUML` plugin. Sections VI and VII discuss plugin development's advantages and shortcomings compared to developing a standalone tool. Finally, Section VIII concludes the paper.

II. Background

A. Model-Driven Engineering

Model-Driven Engineering (MDE) is the software engineering approach that creates increasingly detailed models through transformations, until code is produced. Models are defined according to a metamodel, but the growing number of metamodels led to a higher abstraction level to describe metamodels: meta-metamodels [1]. Arguably the most widely used MDE framework is the OMG's Model-Driven Architecture approach [2], which includes UML (Unified Modeling Language) models [3], OCL (Object Constraint Language) constraints [4] and MOF (Meta-Object Facility) metamodels and meta-metamodels [5].

In the security world, Fernandez-Medina et al. [6] point out that "*current approaches which take security into consideration from the early stages of software development do not take advantage of Model-Driven Development*", but it is a direction that is currently being developed, including by Basin et al. [7], who define Model-Driven Security (MDS) as a specialisation of MDE, where "*a designer builds a system model along with security requirements, and automatically generates from this a complete, configured security infrastructure*".

B. Role-Based Access Control

Traditional access control models [8] allow administrators to assign permission directly to users. This makes the maintenance of large access control directories difficult. By contrast, Role-Based Access Control (RBAC) [9] forbids the direct assignment of permissions to users, and introduces the concept of *roles* between users and permissions. Roles in RBAC are meant to match actual roles in an organisation. Permissions are assigned to roles, which are assigned to users.

The RBAC standard also defines other constructs: role hierarchies, where a role inherits its ancestors' permissions, and static (resp. dynamic) separation of duty, where two roles

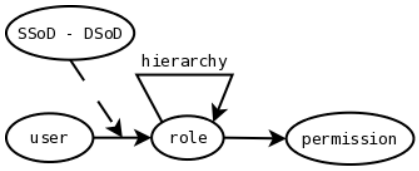


Fig. 1: RBAC model

cannot be assigned to (resp. simultaneously activated by) the same user. Fig. 1 illustrates the RBAC standard.

III. RBACUML

Our `rbacUML` approach integrates RBAC into existing MDE processes. Similar approaches also exist. SecureUML [10], [11] is an application of Basin's MDS allowing designers to model RBAC constraints on UML class diagrams. Another approach, UMLsec [12], allows designers to annotate UML models with any kind of security-related annotations.

`rbacUML` is centered around, but not limited to, two UML profiles for RBAC: one of them allows designers to annotate existing software models with RBAC constructs and requirements: it is the domain-specific modelling annotation language (DSMAL) [13], whilst the other one defines a domain-specific modelling language (DSML) for RBAC only (i.e. it does not allow for the modelling of non-RBAC concepts, as opposed to the DSMAL). Figs. 2a and 2b show a part of a `rbacUML` model. Both profiles are completed by a set of OCL constraints that ensure the well-formedness of models, e.g. that permissions are not directly assigned to users, or that a class is not annotated as both a user and a role. Requirements can be modelled using both profiles, and allow designers to specify properties that the model must enforce. We call them *scenarios*. They can be of two forms: either a requirement that a specific user, given a particular set of roles, must be able to perform a set of actions, or a requirement that a specific user, given a particular set of roles, must *not* be able to perform a set of actions. The evaluation of the conformance of the model to the requirements is done using two OCL constraints. Other constraints complete the profile by providing analysis features such as completeness, model coverage by scenarios, satisfiability, or redundant elements detection.

When a model created using the `rbacUML` DSML is found to violate one or several of the OCL constraints, the plugin is able to suggest solutions that would bring the model to a state where no OCL constraints are violated. This is a very useful feature to users since, often, changing the model to fix a problem in one place is likely to cause other problems, making the search for the best solution difficult and time-consuming.

On large models, evaluating all the OCL constraints on all the model elements can be time consuming. Therefore, we have integrated in `rbacUML` measures to dramatically reduce the evaluation time of models. Other features include a mapping between the `rbacUML` DSML and the LDIF [14] format, an export format used by widely-used LDAP [15] user directories. `rbacUML` is still under development, and will continue to be expanded in order to provide a tight integration of RBAC into existing MDE practices.

IV. CHOOSING THE RIGHT PLATFORM

The choice of developing a plugin for an existing MDE tool, and the choice of the specific modelling tool we used, were not random, nor were they decisions made "on the back of an envelope". In this section we elaborate on the reasons behind those choices and on the alternatives that we have considered. We then elaborate on the architecture of the platform we selected.

A. Plugin rather than Standalone

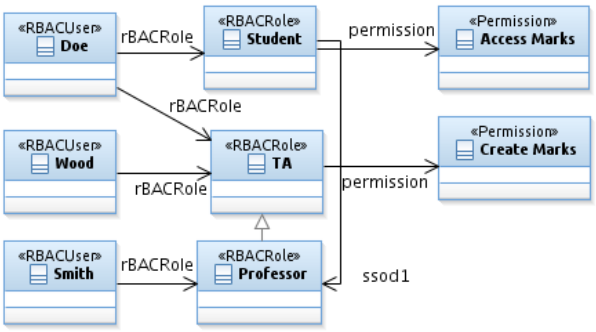
At the core of `rbacUML`, as argued in the previous section, is the desire to integrate `rbacUML` as much as possible into designers' existing activities, processes and tools. Since the DSML part of `rbacUML` is defined as a UML profile, it is no surprise that we greatly valued tightly integrating `rbacUML` with existing UML modelling tools. A plugin seemed to be the right choice, and it was confirmed by a few other considerations. Developing a plugin would allow us to use existing diagramming capabilities, instead of having to implement it ourselves. Since we wanted to use OCL constraints, reusing an existing OCL evaluation engine was also a great argument in favour of a plugin over a standalone tool.

B. Comparing Available Modelling Environments

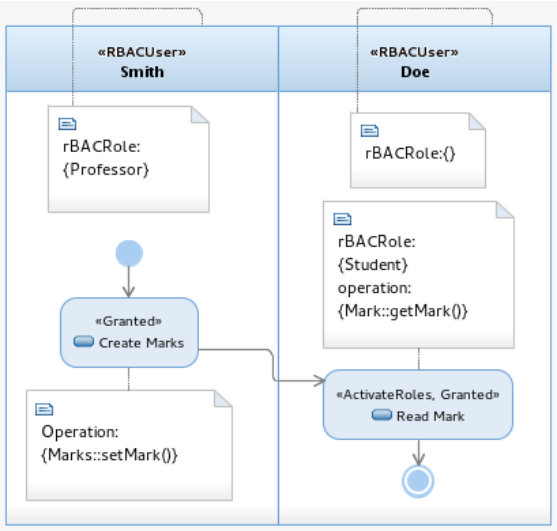
Once it was clear that we were going to develop a plugin instead of a dedicated tool, we had to choose which platform to build it on. We selected three platforms to evaluate, and prepared a list of requirements to compare those platforms:

- UML modelling capabilities: the tool must support the UML 2.x standard;
- OCL queries evaluation engine: since `rbacUML`'s analysis capabilities are based on OCL, an efficient and expressive engine, i.e. a full OCL implementation, is a must;
- support for custom UML profiles: the `rbacUML` DSML is represented as a UML profile, so we had to be able to define our own profile and use it within the platform;
- UML diagrams creation support: models in `rbacUML` are represented using several types of diagrams: class diagrams, sequence diagrams and activity diagrams. It was therefore essential that the platform allows one to easily create at least those three types of diagrams;
- File format: ideally, the perfect tool should use, or at least allow import from and export to, a standardised file format such as the OMG's XML Metadata Interchange (XMI) format, which is an XML extension;
- UML profile tooling generation: an optional, but highly appreciated feature, is the ability to create tooling palettes in order to make the creation of `rbacUML` models easier;
- Licence: an open source tool would be preferable as it would allow for a broader distribution of our plugin.

The three tools we considered, around 2009 - 2010, were Papyrus, an open-source Eclipse plugin for UML modelling [16], ArgoUML, an open-source UML modelling software, and IBM Rational Software Architect (RSA), a proprietary UML modelling solution built on top of Eclipse [17]. Table I



(a) Access Control diagram



(b) Activity diagram

Fig. 2: rbacUML model (notes show associations between different diagrams)

TABLE I: Comparison of MDE environments

Name	UML	OCL	UML profile	UML diagrams	Profile tooling generation	file format	open source
Papyrus (Eclipse)	√	√	partial	buggy	Ø	XMI	√
ArgoUML	1.4 only	partial	no	partial	Ø	zargo & XMI	√
Rational Software Architect	√	√	√	√	√	EMX & XMI	Ø

provides a comparison of the three platforms for each of the requirements that we identified. It is clear from the table that RSA was the only platform that seemed to satisfy most of our requirements. The only issue was that it is proprietary software. It was thus selected.

Papyrus was, at the time, still in its early days. Although it was built on solid Eclipse foundations, such as EMF and its associated OCL evaluation engine, the diagramming part provided by Papyrus was quite slow and buggy. In particular, activity diagrams were very unstable and caused frequent crashes. We had to rule it out, but it has since made a lot of progress.

ArgoUML is the only of the three platforms not to be based on Eclipse. It is a stable product, but only supports UML 1.4, does not support profiles and has a limited OCL support. We had to rule it out as well.

RSA was the last platform we tried. Like Papyrus, it is built on Eclipse and uses EMF and the associated OCL evaluation engine. RSA comes as a layer on top of Eclipse, providing different features, the most notable one being a very mature UML modelling and diagram editing environment. Furthermore, RSA was the only tool allowing us to very easily create tooling palettes for our profile. It uses EMX to store models, an XML format that looks very similar to XMI, and allows for export to and import from XMI. Whilst it isn't an open source project, IBM has an "academic initiative" programme giving academics free access to its products, including RSA.

The RSA platform is built on top of Eclipse, and makes extensive usage of the Eclipse Model Development Tools (MDT) project. It provides many features, but in this section we focus on those directly relevant to our work.

RSA provides a diagram edition layer on top of Eclipse UML, as well as the ability to define UML profiles and automatically generate code for editors that include said profiles. RSA's extensive use of Eclipse MDT technologies makes UML-related projects developed for RSA relatively easy to port to other Eclipse-based tools that also make use of MDT.

V. The rbacUML Plugin

In this section, we brush over the implementation of the `rbacUML` plugin [18], [19], and discuss which Eclipse and/or RSA technologies the plugin uses.

A. The Rational Software Architect Modelling Stack

At the bottom the the UML modelling stack is the Eclipse Modeling Framework (EMF), which uses the Ecore metamodel. On top of EMF is UML2, an EMF implementation of the UML 2.x standard using UML. In other words, the UML metamodel is defined using EMF, and Ecore is therefore used as UML's meta-metamodel.

Eclipse also includes an implementation of an OCL engine, also built on top of EMF, allowing one to parse OCL constraints and use them to evaluate EMF models. In the last few years, the Eclipse project underwent an important change in the implementation of the OCL engine, and two separate implementations co-exist for the duration of the

transition [20]. The mature OCL metamodel provides a parser and an evaluation engine for both Ecore and UML2 model. It is however tightly coupled to Ecore, causing some performance issues and making it difficult to stick to the OCL 2.2 standard. In particular, it makes it very difficult to create and evaluate OCL constraints that work on annotations provided by UML profiles. The new OCL metamodel, called pivot OCL, addresses the shortcomings of the mature OCL metamodel, and complies to the OCL 2.2 standard.

On top of Eclipse MDT comes RSA, which uses the UML modelling and mature OCL metamodel features to provide additional features. The most obvious one is a very efficient UML diagramming capability, allowing one to represent UML2 models as diagram (class, sequence, activity, etc.) instead of "simply" the trees provided by the Eclipse UML2 project. Another feature is the profile tooling generator, that allowed us to easily create UML profiles and automatically generate a RSA plugin to use the profile in UML models.

B. The UML Profile

Building the UML profile was the first step in the tool's implementation. We were able to make use or RSA's profile tooling project capabilities, which allowed us to **(1)** create the stereotypes, specify the types of elements on which they can be attached, and define the associations between the stereotypes, graphically; **(2)** for each stereotype, specify the appropriate OCL constraints; **(3)** generate the tooling model and customise it; and **(4)** generate the tooling code, producing a useable RSA plugin with a tooling palette allowing designers to directly create stereotyped elements.

C. The LDIF import filter

The LDIF import filter, written in Java, is quite simple: it takes an LDIF file as an input, and outputs a UML model. Optionally it can detect and merge duplicate users, i.e. users that have the exact same set of roles. The LDIF format doesn't have a role concept, but uses groups instead. Both users and groups can be members of groups, so groups work very much like roles, and are therefore translated as roles in the output UML model.

For the LDIF filter we decided to create the models in the XMI or EMX formats by directly modifying their XML representation instead of using the EMF helpers. This removes any dependency to Eclipse and RSA. This is especially important as XMI is a file format that is widely used, including by tools such as ArgoUML. Therefore, this part of the plugin can also be used as a standalone tool.

D. OCL Constraints Lazy and Selective Evaluation

Evaluating all OCL constraints on a particular UML model is easy: there's a button for that in the RSA interface. In order to select which constraints to evaluate, however, we had to dig a bit deeper into RSA's API. We used Eclipse's model validation service[1], and its ability to accept *filters* that select which OCL constraints to evaluate. The filters were defined depending on the name of each OCL constraint in the `rbacUML` profile, which always starts with a prefix representing the category in which the constraint falls, e.g. `WF` for well-formedness constraints. The evaluation service is an easy way to select OCL constraints to evaluate, as it uses by default the constraints that are part of the loaded profiles.

[1] `org.eclipse.emf.validation.service.IBatchValidator`

E. The Model Generator

The last feature of the `rbacUML` plugin is the model generator. It has been developed as part of a performance study of the tool. Its purpose is to generate random `rbacUML` models, either correct or incorrect depending on the user's choice, of a specified size. We used it to calculate the evaluation time of increasingly large models, as well as to compare the "full" evaluation of a model with the lazy evaluation.

Unlike the LDIF import filter, the model generator hasn't been implemented by directly generating XML documents. Instead, we made use of Eclipse UML's features, that allow one to very easily create UML model elements - including stereotypes from an existing UML profile.

VI. The Good

There were many advantages in using the Eclipse platform in general, and RSA in particular, to build the `rbacUML` tool.

A. The Profile Builder

Using the UML profile builder to create `rbacUML` proved to be a huge time saver. Indeed, the ability to define the stereotypes and their associations graphically, but also to use the built-in editor to define the OCL constraints, all without writing a single line of non-OCL code, was a much faster way to develop and test our profile than having to manually write the profile as an XMI document. It also made it very easy to come back to the profile to fix a bug, add a new feature or test several alternatives for a particular construct.

B. The Profile Tooling Generator

The profile tooling generator was probably one of RSA's features that saved us the most time. The ability to generate in one click a tooling palette to help designers create `rbacUML` models, and generate a RSA-based environment dedicated to `rbacUML` were incredibly valuable, as the alternative would have been to write all that code manually. The generator also allows for many parameters to be configured before the code is generated, allowing us to tailor the generated tool to our exact needs and requirements.

C. The OCL Engine

To evaluate OCL constraints, we used Eclipse's OCL validation engine, a much better solution than writing our own engine. Eclipse's OCL engine is very powerful and highly configurable. RSA even provides a button to evaluate all the OCL constraints associated to a model in a single click, and when we had to dive deeper into the code to write our own evaluation procedure for selective and lazy evaluations, the OCL engine could be bent to do what we wanted it to do.

D. Creating UML Elements

Programmatically creating UML elements, but also navigating elements through associations, was greatly facilitated by the Eclipse UML component, which does a great job at hiding the underlying complexity of the model. This provided numerous advantages compared to directly editing the EMX files (like we did for the LDIF import filter), or even the EMF or Ecore representations.

E. The Use of Standard Tools and Formats

The fact that both Eclipse and RSA use (mostly) standard technologies and formats was very useful to keep the tool generic enough that it could be ported to other platforms. In fact, the LDIF import filter is even platform-independent: thanks to RSA's usage of the standard XMI format, any tool that also uses XMI can read models created from the filter.

The standard-compliant OCL engine is also worth noting: since it supports the OCL standard, the OCL constraints we wrote can be copied verbatim to another tool with a standard compliant OCL evaluation engine. Furthermore, since RSA uses Eclipse's engine, `rbacUML` should be relatively easy to port to other Eclipse-based tools such as Papyrus.

F. All These Features Come "for Free"

The last advantage of using a plugin, and RSA in particular, are the features that came "for free" and that we didn't have to implement: the diagramming capability; the error reporting, in the tree-like model explorer, on the diagrams themselves, *and* in a textual form in a dedicated view; etc.

It would be almost impossible to list all the features of the platform that saved us time. In this section, we have pointed out the most salient positive points.

VII. The Bad, and the Ugly

Now we focus on the less positive parts of the development of `rbacUML` - things that didn't go very well, blocking bugs or difficulties that we encountered. This section is not meant to be understood as criticism towards the tools we used or the team behind them, but instead, it points plugin developers to areas they need to pay particular attention to, where they are likely to encounter difficulties, and it provides the platform developers with pointers on how to improve the platform to make third-party developers' work easier.

A. The Profile Tooling Generator

Whilst the profile tooling generator saved us huge amounts of time, it isn't perfect, and there were situations where we had to dive into the generated code to fix some issues.

The first issue was a bug in the code generation of stereotypes attached to `Action` elements. One of the subtypes of the `Action` type in UML was causing the generated code to produce a very unstable tool. Fortunately the stereotypes applied on `Action` elements didn't really need to be applied on that particular subtype, and we could simply remove it from the list of elements on which the stereotypes could be applied - and regenerate the tooling code.

The second issue was not a bug, but had to do with the way the code generator works, and with the incremental way we developed the UML profile. Usually we were adding new elements at each iteration, but occasionally we had to delete elements as well, as we realised that a particular RBAC construct could be better or more succinctly expressed with another construct. The tooling code generator works in a quite conservative way, to make sure that user-defined code isn't overwritten unless necessary. Therefore, re-running the generator after it has already run at least once will not result in the generator wiping out the existing code and replacing it by its own (thankfully!), but instead, it will only overwrite files that it generates, and leave the others alone. This means that, if an element is *removed*, the implementation of the related features will be left in the code (since they are not generated anymore), and cause compilation problems. It is then the users' responsibility to go through all the compilation errors and remove the now useless classes and references to elements that do not exist anymore. This is a problem we are trying to solve using bidirectional transformations to synchronise user changes with the generated models [21].

B. Bugs

We encountered a few annoying bugs in the platform, that forced us to develop workarounds and/or dive into the lower layers of the platform. In particular, bugs in the OCL evaluation engine made it more difficult to navigate associations between stereotypes. Furthermore, parts of the evaluation engine couldn't actually deal with OCL queries that returned non-boolean values, even though the documentation indicates otherwise. This prompted us to rewrite these queries so they would return boolean values, or to use a lower level of abstraction to get around the problem.

There were also bugs that made it impossible to navigate stereotype associations in Java using the UML abstraction level, and we had to use the underlying representation.

C. The Size of the Platform

The Eclipse platform is *huge*, and so is the RSA platform. Combined, they form a gargantuan set of technologies, built to work on top of or in combination with each other. While this obviously provides immense benefits, it also comes with its faults and weaknesses. It can be very difficult and intimidating for developers that are new to the platform to get a working understanding of how all the pieces fit together. RSA includes some documentation, which frequently refers to the Eclipse documentation, but dead links are not uncommon. Furthermore, the IBM academic initiative does not include any support from IBM or rational, so we were left on our own to figure out how the platform works and what its limitation are, only with the help of the documentation, which is often incomplete. We also used the community support, via the IBM forums or websites such as Stackoverflow [22], but got very few (if any) answers on some of the most advanced questions. It seems that there isn't a massive community of advanced OCL users, or if it exists, we have yet to find it.

Learning how to use the platform requires a large time investment, especially for developers that do not have an Eclipse/RSA expert handy. We had to learn about the Eclipse platform, about plugin development for Eclipse, about the Eclipse MDT project and its limitations, and about RSA. We then had to put all that information together and figure out how these projects relate to each other. It took us months, and we are still learning every day. We are documenting our experience to make it easier for the community to develop similar modelling plugins for the Eclipse or RSA platforms.

D. Limitations to Dissemination

Our choice of RSA as a platform did most probably limit the potential for dissemination of `rbacUML`. Indeed, whilst RSA is available for free to academics, few are willing to make the effort to deploy it, most notably because of the rather large amount of online paperwork to fill, the obligations that come together with the IBM academic initiative, and the lack of support for Mac OS X (although there is now a preview version of RSA 8.5 for Mac). Non-academics were understandably reluctant to invest in costly RSA licence fees, which made it very difficult to reach out to industry.

In order to mitigate this issue, we aim to avoid using RSA-specific APIs as much as possible, and instead try to rely on Eclipse MDT alone whenever possible. We have been much more successful at this with our most recent developments, and one of the added benefits is that there is less documentation to deal with. While currently RSA is still required to run our plugin, we hope that, by further diminishing our reliance on IBM's proprietary APIs, and thanks to the progress of open-source tools such as Papyrus, we will be able to migrate to a fully open source platform in the near future, which will doubtlessly make it easier for third parties to use, and perhaps contribute to or build upon our plugin.

VIII. Conclusion

In this paper, we have discussed the implementation of our approach to integrate RBAC concerns in an MDE process. The choice of implementing our tool as a plugin of an existing MDE platform was quite straightforward as it allowed for a very tight integration of our approach with existing practice. We have reported on our criteria to select a modelling plugin platform, and on the good, as well as the less good, things that we encountered. Although our experience is based on modelling with UML in Eclipse and RSA, similar criteria and issues may certainly apply to other modelling languages and platforms. We derive the following suggestions to individuals or organisations willing to take on a similar route.

- **Plugins** are definitely the way to go to achieve excellent integration very quickly. The amount of time saved by the ability to reuse existing components is perhaps the best argument in favour of using a plugin;
- If interoperability is a concern, one will be very careful about the platform's support of **standards**;
- Sufficient **time** will have to be allocated to acquire in-depth knowledge of the platform. Even the lower layers may have to be used to get around bugs and problems;
- If the tool is meant to be used by a large public, the platform must be carefully chosen to make it as easy as possible to adopt.

Overall, the `rbacUML` experience has been positive, and further development is under way to integrate the `rbacUML` approach even more with designers' MDE workflow.

References

[1] J. Bézivin, F. Jouault, and D. Touzet, "Principles, standards and tools for model engineering," in *ICECCS: Procs. Intl. Conf. on Engineering of Complex Computer Systems*. IEEE, 2005, pp. 28–29.

[2] R. Soley and the OMG staff, "Model driven architecture," white paper, November 2000, last accessed 14 June 2010. [Online]. Available: http://www.omg.org/cgi-bin/doc?omg/00-11-05

[3] *Unified Modeling Language (UML) 2.3*, OMG Std.

[4] *Object Constraint Language 2.2*, OMG Std.

[5] OMG, *Meta Object Facility (MOF) 2.0*, OMG Std.

[6] E. Fernández-Medina, J. Jurjens, J. Trujillo, and S. Jajodia, "Model-driven development for secure information systems," *Information and Software Technology*, vol. 51, no. 5, pp. 809 – 814, 2009, sPECIAL ISSUE: Model-Driven Development for Secure Information Systems.

[7] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security for process-oriented systems," in *SACMAT: Procs. Symposium on Access Control Models and Technologies*. ACM, 2003, pp. 100–109.

[8] M. H. Klein, *Department of Defense Trusted Computer System Evaluation Criteria*, Department of Defense Std. CSC-STD-001-83, 1983.

[9] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, 2001.

[10] D. Basin, J. Doser, and T. Lodderstedt, "Model driven security: From UML models to access control infrastructures," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, no. 1, pp. 39–91, 2006.

[11] D. A. Basin, M. Clavel, and M. Egea, "A decade of model-driven security," in *SACMAT: Procs. Symposium on Access Control Models and Technologies*, 2011, pp. 1–10.

[12] J. Jürjens, *Secure Systems Development with UML*. Springer-Verlag, 2005.

[13] L. Montrieux, Y. Yu, M. Wermelinger, and Z. Hu, "Issues in representing domain-specific concerns in model-driven engineering," in *MiSE: Procs. Workshop on Modeling in Software Engineering*. IEEE, 2013.

[14] IETF Network Working Group, "RFC 2849 - The LDAP Data Interchange Format (LDIF) - Technical Specification," 2000. [Online]. Available: https://www.ietf.org/rfc/rfc2849.txt

[15] IETF, *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map (RFC 4510)*, IETF Std.

[16] S. Gérard, C. Dumoulin, P. Tessier, and B. Selic, "19 Papyrus: A UML2 tool for domain-specific language modeling," in *Model-Based Engineering of Embedded Real-Time Systems*, ser. Lecture Notes in Computer Science, vol. 6100. Springer, 2011, pp. 361–368.

[17] IBM, "Rational Software Architect 8.0.4," 2012.

[18] L. Montrieux, M. Wermelinger, and Y. Yu, "Tool support for UML-based specification and verification of role-based access control properties," in *ESEC/FSE: Procs. SIGSOFT Symposium and European Conf. on Foundations of Software Engineering*. ACM, 2011, pp. 456–459.

[19] "rbacUML tool," 2009-2012. [Online]. Available: http://computing-research.open.ac.uk/rbac/

[20] Eclipse Foundation, "OCL users guide," last accessed February 2013. [Online]. Available: http://goo.gl/zdIB9

[21] Y. Yu, Y. Lin, Z. Hu, S. Hidaka, H. Kato, and L. Montrieux, "Maintaining invariant traceability through bidirectional transformations," in *ICSE: Procs. Intl. Conf. on Software Engineering*, 2012, pp. 540–550.

[22] Stack Exchange, inc., "Stackoverflow," last accessed March 2013. [Online]. Available: http://www.stackoverflow.com